Comparing Information Extraction Pattern Models

Mark Stevenson and Mark A. Greenwood
Department of Computer Science
University of Sheffield
Sheffield, S1 4DP, UK
{marks,m.greenwood}@dcs.shef.ac.uk

Abstract

Several recently reported techniques for the automatic acquisition of Information Extraction (IE) systems have used dependency trees as the basis of their extraction pattern representation. These approaches have used a variety of pattern models (schemes for representing IE patterns based on particular parts of the dependency analysis). An appropriate model should be expressive enough to represent the information which is to be extracted from text without being overly complicated. Four previously reported pattern models are evaluated using existing IE evaluation corpora and three dependency parsers. It was found that one model, linked chains, could represent around 95% of the information of interest without generating an unwieldy number of possible patterns.

1 Introduction

A common approach to Information Extraction (IE) is to use patterns which match against text and identify items of interest. Patterns are applied to text which has undergone various levels of linguistic analysis, such as phrase chunking (Soderland, 1999) and full syntactic parsing (Gaizauskas et al., 1996). The approaches use different definitions of what constitutes a valid pattern. For example, the AutoSlog system (Riloff, 1993) uses patterns which match certain grammatical categories, mainly nouns and verbs, in phrase chunked text while Yangarber et al. (2000) use subject-verb-object tuples derived from a dependency parse. An appropriate pattern language must encode enough information about the text to be able to accurately identify the items of interest. However, it should not contain so much information as to be complex and impractical to apply.

Several recent approaches to IE have used patterns based on a dependency analysis of the input text (Yangarber, 2003; Sudo et al., 2001; Sudo et al., 2003; Bunescu and Mooney, 2005; Stevenson and Greenwood, 2005). These approaches have used a variety of pattern models (schemes for representing IE patterns based on particular parts of the dependency tree). For example, Yangarber (2003) uses just subject-verb-object tuples while Sudo et al. (2003) allow any subpart of the tree to act as an extraction pattern. The set of patterns allowed by the first model is a proper subset of the second and therefore captures less of the information contained in the dependency tree. Little analysis has been carried out into the appropriateness of each model. Sudo et al. (2003) compared three models in terms of their ability to identify event participants.

The choice of pattern model has an effect on the number of potential patterns. This has implications on the practical application for each approach, particularly when used for automatic acquisition of IE systems using learning methods (Yangarber et al., 2000; Sudo et al., 2003; Bunescu and Mooney, 2005). This paper evaluates the appropriateness of four pattern models in terms of the competing aims of expressive completeness (ability to represent information in text) and complexity (number of possible patterns). Each model is examined by comparing it against a corpus annotated with events and determining the proportion of those which it is capable of representing.

The remainder of this paper is organised as follows: a variety of dependency-tree-based IE pat-

Proceedings of the Workshop on Information Extraction Beyond The Document, pages 12–19,
Sydney, July 2006.

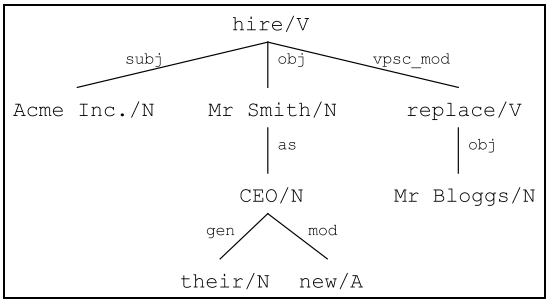

Figure 1: An example dependency tree.

tern models are introduced (Sections 2 and 3). Section 4 describes experiments comparing each model and the results are discussed in Section 5.

2 Pattern Models

In dependency analysis (Mel'čuk, 1987) the syntax of a sentence is represented by a set of directed binary links between a word (the head) and one of its modifiers. These links may be labelled to indicate the grammatical relation between the head and modifier (e.g. subject, object). In general cyclical paths are disallowed so that the analysis forms a tree structure. An example dependency analysis for the sentence *"Acme Inc. hired Mr Smith as their new CEO, replacing Mr Bloggs."* is shown Figure 1.

The remainder of this section outlines four models for representing extraction patterns which can be derived from dependency trees.

Predicate-Argument Model (SVO): A simple approach, used by Yangarber (2003) and Stevenson and Greenwood (2005), is to use subject-verb-object tuples from the dependency parse as extraction patterns. These consist of a verb and its subject and/or direct object[1]. An SVO pattern is extracted for each verb in a sentence. Figure 2 shows the two SVO patterns[2] which are produced for the dependency tree shown in Figure 1.

This model may be motivated by the assumption that many IE scenarios involve the extraction of participants in specific events. For example, the MUC-6 (MUC, 1995) management succession scenario concerns the identification of individuals who are changing job. These events are often described using a simple predicate argument structure, e.g. *"Acme Inc. fired Smith"*. However, the SVO model cannot represent information described using other linguistic constructions such as nominalisations or prepositional phrases. For example, in the MUC6 texts it is common for job titles to be mentioned within prepositional phrases, e.g. *"Smith joined Acme Inc. as CEO"*.

Chains: A pattern is defined as a path between a verb node and any other node in the dependency tree passing through zero or more intermediate nodes (Sudo et al., 2001). Figure 2 shows the eight chains which can be extracted from the tree in Figure 1.

Chains provide a mechanism for encoding information beyond the direct arguments of predicates and includes areas of the dependency tree ignored by the SVO model. For example, they can represent information expressed as a nominalisation or within a prepositional phrase, e.g. *"The resignation of Smith from the board of Acme ..."* However, a potential shortcoming of this model is that it cannot represent the link between arguments of a verb. Patterns in the chain model format are unable to represent even the simplest of sentences containing a transitive verb, e.g. *"Smith left Acme Inc."*.

Linked Chains: The linked chains model (Greenwood et al., 2005) represents extraction patterns as a pair of chains which share the same verb but no direct descendants. This model generates 14 patterns for the verb *hire* in Figure 1, examples of which are shown in Figure 2. This pattern representation encodes most of the information in the sentence with the advantage of being able to link together event participants which neither of the SVO or chain model can, for example the relation between *"Smith"* and *"Bloggs"*.

Subtrees: The final model to be considered is the subtree model (Sudo et al., 2003). In this model any subtree of a dependency tree can be used as an extraction pattern, where a subtree is any set of nodes in the tree which are connected to one another. Single nodes are not considered to be subtrees. The subtree model is a richer representation than those discussed so far and can represent any part of a dependency tree. Each of the previ-

[1] Yangarber et al. (2000) and Sudo et al. (2003) used a slightly extended version of this model in which the pattern also included certain phrases which referred to either the subject or object.

[2] The formalism used for representing dependency patterns is similar to the one introduced by Sudo et al. (2003). Each node in the tree is represented in the format `a[b/c]` (e.g. `subj[N/bomber]`) where `c` is the lexical item (`bomber`), `b` its grammatical tag (`N`) and `a` the dependency relation between this node and its parent (`subj`). The relationship between nodes is represented as `X(A+B+C)` which indicates that nodes `A`, `B` and `C` are direct descendents of node `X`.

SVO
```
[V/hire](subj[N/Acme Inc.]+obj[N/Mr Smith])
[V/replace](obj[N/Mr Bloggs])
```

Chains
```
[V/hire](subj[N/Acme Inc.])
[V/hire](obj[N/Mr Smith])
[V/hire](obj[N/Mr Smith](as[N/CEO]))
[V/hire](obj[N/Mr Smith](as[N/CEO](gen[N/their])))
[V/hire](obj[N/Mr Smith](as[N/CEO](mod[A/new])))
[V/hire](vpsc_mod[V/replace])
[V/hire](vpsc_mod[V/replace](obj[N/Mr Bloggs]))
[V/replace](obj[N/Mr Bloggs])
```

Linked Chains
```
[V/hire](subj[N/Acme Inc.]+obj[N/Mr Smith])
[V/hire](subj[N/Acme Inc.]+obj[N/Mr Smith](as[N/CEO]))
[V/hire](obj[N/Mr Smith]+vpsc_mod[V/replace](obj[N/Mr Bloggs]))
```

Figure 2: Example patterns for three models

ous models form a proper subset of the subtrees. By choosing an appropriate subtree it is possible to link together any pair of nodes in a tree and consequently this model can represent the relation between any set of items in the sentence.

3 Pattern Enumeration and Complexity

In addition to encoding different parts of the dependency analysis, each pattern model will also generate a different number of potential patterns.

A dependency tree, T, can be viewed as a set of N connected nodes. Assume that V, such that $V \subseteq N$, is the set of nodes in the dependency tree labelled as a verb.

Predicate-Argument Model (SVO): The number of SVO patterns extracted from T is:

$$N_{svo}(T) = |V| \tag{1}$$

Chain Model: A chain can be created between any verb and a node it dominates (directly or indirectly). Now assume that $d(v)$ denotes the count of a node v and all its descendents then the number of chains is given by:

$$N_{chains}(T) = \sum_{v \in V} (\, d(v) - 1 \,) \tag{2}$$

Linked Chains: Let $C(v)$ denote the set of direct child nodes of node v and v_i denote the i-th child, so $C(v) = \{v_1, v_2, ...v_{|C(v)|}\}$. The number of possible linked chains in T is given by:

$$N_{linked\,chains}(T) = \sum_{v \in V} \sum_{i=1}^{|C(v)|} \sum_{j=i+1}^{|C(v)|} d(v_i)\, d(v_j) \tag{3}$$

Subtrees: Now assume that $sub(n)$ is a function denoting the number of subtrees, including single nodes, rooted at node n. This can be defined recursively as follows:

$$sub(n) = \begin{cases} 1 & \text{if } n \text{ is a leaf node} \\ \prod_{i=1}^{|C(n)|} (sub(n_i) + 1) & \text{otherwise} \end{cases} \tag{4}$$

The total number of subtrees in a tree is given by:

$$N_{subtree}(T) = \left(\sum_{n \in N} sub(n) \right) - |N| \tag{5}$$

The dependency tree shown in Figure 1 generates 2, 8, 14 and 42 possible SVO, chain, linked chain and subtree patterns respectively. The number of SVO patterns is constant on the number of verbs in the tree. The number of chains is generally a linear function on the size of the tree but, in the worst case, can be polynomial. The linked chain model generates a polynomial number of patterns while the subtree model is exponential.

There is a clear tradeoff between the complexity of pattern representations and the practicality of computation using them. Some pattern representations are more expressive, in terms of the amount of information from the dependency tree they make use of, than others (Section 2) and are therefore more likely to produce accurate extraction patterns. However, the more expressive models will add extra complexities during computation since a greater number of patterns will be generated. This complexity, both in the number of patterns produced and the computational effort required to produce them, limits the algorithms that can reasonably be applied to learn useful extraction patterns.

For a pattern model to be suitable for an extraction task it needs to be expressive enough to encode enough information from the dependency parse to accurately identify the items which need to be extracted. However, we also aim for the

model to be as computationally tractable as possible. The ideal model will then be one with sufficient expressive power while at the same time not including extra information which would make its use less practical.

4 Experiments

We carried out experiments to determine how suitable the pattern representations detailed in Section 2 are for encoding the information of interest to IE systems. We chose a set of IE corpora annotated with the information to be extracted (detailed in Section 4.1), generated sets of patterns using a variety of dependency parsers (Section 4.2) which were then examined to discover how much of the target information they contain (Section 4.3).

4.1 Corpora

Corpora representing different genres of text were chosen for these experiments; one containing newspaper text and another composed of biomedical abstracts. The first corpus consisted of Wall Street Journal texts from the Sixth Message Understanding Conference (MUC, 1995) IE evaluation. These are reliably annotated with details about the movement of executives between jobs. We make use of a version of the corpus produced by Soderland (1999) in which events described within a single sentence were annotated. Events in this corpus identify relations between up to four entities: `PersonIn` (the person starting a new job), `PersonOut` (person leaving a job), `Post` (the job title) and `Organisation` (the employer). These events were broken down into a set of binary relationships. For example, the sentence *"Smith was recently made chairman of Acme."* contains information about the new employee (*Smith*), post (*chairman*) and organisation (*Acme*). Events are represented as a set of binary relationships, `Smith-chairman`, `chairman-Acme` and `Smith-Acme` for this example.

The second corpus uses documents taken from the biomedical domain, specifically the training corpus used in the LLL-05 challenge task (Nédellec, 2005), and a pair of corpora (Craven and Kumlien, 1999) which were derived from the Yeast Proteome Database (YPD) (Hodges et al., 1999) and the Online Mendelian Inheritance in Man database (OMIM) (Hamosh et al., 2002). Each of these corpora are annotated with binary relations between pairs of entities. The LLL-05 corpora contains interactions between genes and proteins. For example the sentence *"Expression of the sigma(K)-dependent cwlH gene depended on gerE"* contains relations between *sigma(K)* and *cwlH* and between *gerE* and *cwlH*. The YPD corpus is concerned with the subcellular compartments in which particular yeast proteins localize. An example sentence *"Uba2p is located largely in the nucleus"* relates *Uba2p* and *the nucleus*. The relations in the OMIM corpora are between genes and diseases, for example *"Most sporadic colorectal cancers also have two APC mutations"* contains a relation between *APC* and *colorectal cancer*.

The MUC6 corpus contains a total of six possible binary relations. Each of the three biomedical corpora contain a single relation type, giving a total of nine binary relations for the experiments. There are 3911 instances of binary relations in all corpora.

4.2 Generating Dependency Patterns

Three dependency parsers were used for these experiments: MINIPAR[3] (Lin, 1999), the Machinese Syntax[4] parser from Connexor Oy (Tapanainen and Järvinen, 1997) and the Stanford[5] parser (Klein and Manning, 2003). These three parsers represent a cross-section of approaches to producing dependency analyses: MINIPAR uses a constituency grammar internally before converting the result to a dependency tree, Machinese Syntax uses a functional dependency grammar, and the Stanford Parser is a lexicalized probabilistic parser.

Before these parsers were applied to the various corpora the named entities participating in relations are replaced by a token indicating their class. For example, in the MUC6 corpus *"Acme hired Smith"* would become "`Organisation` *hired* `PersonIn`". Each parser was adapted to deal with these tokens correctly. The parsers were applied to each corpus and patterns extracted from the dependency trees generated.

The analyses produced by the parsers were post-processed to make the most of the information they contain and ensure consistent structures from which patterns could be extracted. It was found

[3] `http://www.cs.ualberta.ca/~lindek/`
[4] `http://www.connexor.com/software/syntax/`
[5] `http://www-nlp.stanford.edu/software/`

Parser	SVO	Chains	Linked chains	Subtrees
MINIPAR	2,980	52,659	149,504	353,778,240,702,149,000
Machinese Syntax	2,382	67,690	265,631	4,641,825,924
Stanford	2,950	76,620	478,643	1,696,259,251,073

Table 1: Number of patterns produced for each pattern model by different parsers

that the parsers were often unable to generate a dependency tree which included the whole sentence and instead generate an analysis consisting of sentence fragments represented as separate tree structures. Some fragments did not include a verb so no patterns could be extracted. To take account of this we allowed the root node of any tree fragment to take the place of a verb in a pattern (see Section 2). This leads to the generation of more chain and linked chain patterns but has no effect on the number of SVO patterns or subtrees.

Table 1 shows the number of patterns generated from the dependency trees produced by each of the parsers. The number of subtrees generated from the MINIPAR parses is several orders of magnitude higher than the others because MINIPAR allows certain nodes to be the modifier of two separate nodes to deal with phenomena such as conjunction, anaphora and VP-coordination. For example, in the sentence *"The bomb caused widespread damage and killed three people" the bomb* is the subject of both the verbs *cause* and *kill*. We made use of this information by duplicating any nodes (and their descendants) with more than one head.[6]

Overall the figures in Table 1 are consistent with the analysis in Section 3 but there is great variation in the number of patterns produced by the different parsers. For example, the Stanford parser produces more chains and linked chains than the other parsers. (If we did not duplicate portions of the MINIPAR parses then the Stanford parser would also generate the most subtrees.) We found that the Stanford parser was the most likely to generate a single dependency tree for each sentence while the other two produced a set of tree fragments. A single dependency analysis contains a greater number of patterns, and possible subtrees, than a fragmented analysis. One reason for this may be that the Stanford parser is unique in allowing the use of an underspecified dependency relation, `dep`, which can be applied when the role of the dependency is unclear. This allows the Stanford parser to generate analyses which span more of the sentence than the other two.

4.3 Evaluating Pattern Models

Patterns from each of the four models are examined to check whether they cover the information which should be extracted. In this context "cover" means that the pattern contains both elements of the relation. For example, an SVO pattern extracted from the dependency parse of *"Smith was recently made chairman of Acme."* would be `[V/make](subj[N/Smith]+obj[N/chairman])` which covers the relation between *Smith* and *chairman* but not the relations between *Smith* and *Acme* or *chairman* and *Acme*. The coverage of each model is computed as the percentage of relations in the corpus for which at least one of the patterns contains both of the participating entities. Coverage is related to the more familiar IE evaluation metric of recall since the coverage of a pattern model places an upper bound on the recall of any system using that model. The aim of this work is to determine the proportion of the relations in a corpus that can be represented using the various pattern models rather than their performance in an IE system and, consequently, we choose to evaluate models in terms of their coverage rather than precision and recall.[7]

For practical applications parsers are required to generate the dependency analysis but these may not always provide a complete analysis for every sentence. The coverage of each model is influenced by the ability of the parser to produce a tree which connects the elements of the event to be extracted. To account for this we compute the coverage of each model relative to a particular parser. The subtree model covers all events whose entities are included in the dependency tree and, consequently, the coverage of this model represents the maximum number of events that the model can

[6] One dependency tree produced by MINIPAR, expanded in this way, contained approximately 1×10^{64} subtrees. These are not included in the total number of subtrees for the MINIPAR parses shown in the table.

[7] The subtree model can be used to cover any set of items in a dependency tree. So, given accurate dependency analyses, this model will cover all events. The coverage of the subtree model can be determined by checking if the elements of the event are connected in the dependency analysis of the sentence and, for simplicity, we chose to do this rather than enumerating all subtrees.

represent for a given dependency tree. The coverage of other models relative to a dependency analysis can be computed by dividing the number of events it covers by the number covered by the subtree model (i.e. the maximum which can be covered). This measure is refered to as the bounded coverage of the model. Bounded coverage for the subtree model is always 100%.

5 Results

Coverage and bounded-coverage results for each pattern representation and parser combination are given in Table 2. The table lists the corpus, the total number of instances within that corpus and the results for each of the four pattern models. Results for the subtree model lists the coverage and raw count, the bounded-coverage for this model will always be 100% and is not listed. Results for the other three models show the coverage and raw count along with the bounded coverage. The coverage of each parser and pattern representation (combined across both corpora) are also summarised in Figure 3.

The simplest representation, SVO, does not perform well in this evaluation. The highest bounded-coverage score is 15.1% (MUC6 corpus, Stanford parser) but the combined average over all corpora is less than 6% for any parser. This suggests that the SVO representation is simply not expressive enough for IE. Previous work which has used this representation have used indirect evaluation: document and sentence filtering (Yangarber, 2003; Stevenson and Greenwood, 2005). While the SVO representation may be expressive enough to allow a classifier to distinguish documents or sentences which are relevant to a particular extraction task it seems too limited to be used for relation extraction. The SVO representation performs noticeably worse on the biomedical text. Our analysis suggests that this is because the items of interest are commonly described in ways which the SVO model is unable to represent.

The more complex chain model covers a greater percentage of the relations. However its bounded-coverage is still less than half of the relations in either the MUC6 corpus or the biomedical texts. Using the chain model the best coverage which can be achieved over any corpus is 41.07% (MUC6 corpus, MINIPAR and Stanford parser) which is unlikely to be sufficient to create an IE system.

Results for the linked chain representation are much more promising covering around 70% of all relations using the MINIPAR and Machinese Syntax parsers and over 90.64% using the Stanford parser. For all three parsers this model achieves a bounded-coverage of close to 95%, indicating that this model can represent the majority of relations which are included in a dependency tree. The subtree representation covers slight more of the relations than linked chains: around 75% using the MINIPAR or Machinese Syntax parsers and 96.62% using the Stanford parser.

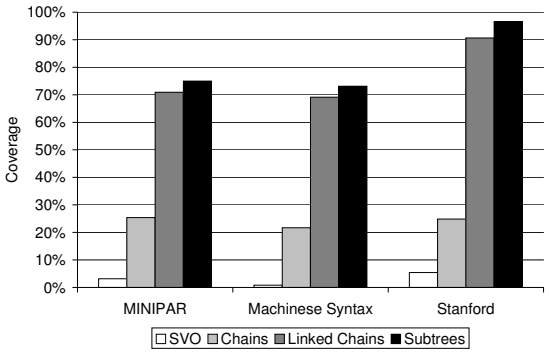

Figure 3: Coverage of various pattern representation models for each of the three parsers.

A one-way repeated measures ANOVA was carried out to analyse the differences between the results for each model shown in Table 2. It was found that the differences between the SVO, chain, linked chain and subtree models are significant ($p < 0.01$). A Tukey test was then applied to identify which of the individual differences between pairs of models were significant. Differences between two pairs of models were not found to be significant ($p < 0.01$): SVO and chains; linked chains and subtrees.

These results suggest that the linked chains and subtree models can represent significantly more of the relations which occur in IE scenarios than either the SVO or chain models. However, there is little to be gained from using the subtree model since accuracy of the linked chain model is comparable and the number of patterns generated is bounded by a polynomial rather than exponential function.

5.1 Analysis and Discussion

Examination of the relations which were covered by the subtree model but not by linked chains suggested that there are certain constructions which cause difficulties. One such construction is the appositive, e.g. the relation between

Parser	Corpus	# of Relations	SVO %C	SVO %B-C	Chains %C	Chains %B-C	Linked Chains %C	Linked Chains %B-C	Subtrees %C
MINIPAR	MUC6	1322	7.49 (99)	9.07	41.07 (543)	49.73	81.92 (1083)	99.18	82.60 (1092)
	Biomed	2589	0.93 (24)	1.30	17.38 (450)	24.44	65.31 (1691)	91.85	71.11 (1841)
	Combined	3911	3.14 (123)	4.19	25.39 (993)	33.86	70.93 (2774)	94.58	74.99 (2933)
Machinese Syntax	MUC6	1322	2.12 (28)	2.75	35.70 (472)	46.41	76.32 (1009)	99.21	76.93 (1017)
	Biomed	2589	0.19 (5)	0.27	14.56 (377)	20.47	65.47 (1695)	92.02	71.15 (1842)
	Combined	3911	0.84 (33)	1.15	21.71 (849)	29.70	69.14 (2704)	94.58	73.10 (2859)
Stanford	MUC6	1322	15.05 (199)	15.10	41.07 (543)	41.20	94.78 (1253)	95.07	99.70 (1318)
	Biomed	2589	0.46 (12)	0.49	16.53 (428)	17.39	88.52 (2292)	93.13	95.06 (2461)
	Combined	3911	5.40 (211)	5.58	24.83 (971)	25.69	90.64 (3545)	93.81	96.62 (3779)

Table 2: Evaluation results for the three different parsers.

`PersonOut` and `Organisation` in the fragment "`Organisation`*'s* `Post`, `PersonOut`, *resigned yesterday morning*". Certain nominalisations may also cause problems for the linked chains representation, e.g. in biomedical text the relation between `Agent` and `Target` in the nominalisation "*the* `Agent`*-dependent assembly of* `Target`" cannot be represented by a linked chain. In both cases the problem is caused by the fact that the dependency tree generated includes the two named entities in part of the tree dominated by a node marked as a noun. Since each linked chain must be anchored at a verb (or the root of a tree fragment) and the two chains cannot share part of their path, these relations are not covered. It would be possible to create another representation which allowed these relations to be captured but it would generate more patterns than the linked chain model.

Our results also reveal that the choice of dependency parser effects the coverage of each model (see Figure 3). The subtree model coverage scores for each parser shown in Table 3 represent the percentage of sentences for which an analysis was generated that included both items from the binary relations. These figures are noticably higher for the Stanford parser. We previously mentioned (Section 4.2) that this parser allows the use of an underspecified dependency relation and suggested that this may be a reason for the higher coverage. The use of underspecified dependency relations may not be useful for all applications but is unlikely to cause problems for systems which learn IE patterns provided the trees generated by the parser are consistent. Differences between the results produced by the three parsers suggest that it is important to fully evaluate their suitability for a particular purpose.

These experiments also provide insights into the more general question of how suitable dependency trees are as a basis for extraction patterns. Dependency analysis has the advantage of generating analyses which abstract away from the surface realisation of text to a greater extent than phrase structure grammars tend to. This leads to the semantic information being more accessible in the representation of the text which can be useful for IE. For practical applications this approach relies on the ability to accurately generate dependency analyses. The results presented here suggest that the Stanford parser (Klein and Manning, 2003) is capable of generating analyses for almost all sentences within corpora from two very different domains. Bunescu and Mooney (2005) have also demonstrated that dependency graphs can be produced using Combinatory Categorial Grammar (CCG) and context-free grammar (CFG) parsers.

6 Conclusions

This paper compares four IE pattern models: SVO, chains, linked chains and subtrees. Using texts from the management succession and biomedical domains it was found that the linked chains model can represent around 95% of the possible relations contained in the text, given a dependency parse. Subtrees can represent all the relations contained within dependency trees but their use is less practical because enumerating all possible subtrees is a more complex problem and the large number of resulting patterns could limit the learning algorithms that can be applied. This result should be borne in mind during the design of IE systems.

Acknowledgements

The authors are grateful to Mike Stannet for providing the method for counting subtrees introduced in Section 3 and to Connexor Oy for use of the Machinese Syntax parser. The research

described in this paper was funded by the Engineering and Physical Sciences Research Council via the RESuLT project (GR/T06391) and partially funded by the IST 6th Framework project X-Media (FP6-26978).

References

Razvan Bunescu and Raymond Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 724–731, Vancouver, B.C.

Mark Craven and Johan Kumlien. 1999. Constructing Biological Knowledge Bases by Extracting Information from Text Sources. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 77–86, Heidelberg, Germany. AAAI Press.

Robert Gaizauskas, Takahiro Wakao, Kevin Humphreys, Hamish Cunningham, and Yorick Wilks. 1996. Description of the LaSIE system as used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pages 207–220, San Francisco, CA.

Mark A. Greenwood, Mark Stevenson, Yikun Guo, Henk Harkema, and Angus Roberts. 2005. Automatically Acquiring a Linguistically Motivated Genic Interaction Extraction System. In *Proceedings of the 4th Learning Language in Logic Workshop (LLL05)*, Bonn, Germany.

Ada Hamosh, Alan F. Scott, Joanna Amberger, Carol Bocchini, David Valle, and Victor A. McKusick. 2002. Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders. *Nucleic Acids Research*, 30(1):52–55.

Peter E. Hodges, Andrew H. Z. McKee, Brian P. Davis, William E. Payne, and James I. Garrels. 1999. The Yeast Proteome Database (YPD): a model for the organization and presentation of genome-wide functional data. *Nucleic Acids Research*, 27(1):69–73.

Dan Klein and Christopher D. Manning. 2003. Accurate Unlexicalized Parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*, pages 423–430, Sapporo, Japan.

Dekang Lin. 1999. MINIPAR: A Minimalist Parser. In *Maryland Linguistics Colloquium*, University of Maryland, College Park.

Igor Mel'čuk. 1987. *Dependency Syntax: Theory and Practice*. SUNY Press, New York.

MUC. 1995. *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, San Mateo, CA. Morgan Kaufmann.

Claire Nédellec. 2005. Learning Language in Logic - Genic Interaction Extraction Challenge. In *Proceedings of the 4th Learning Language in Logic Workshop (LLL05)*, Bonn, Germany, August.

Ellen Riloff. 1993. Automatically constructing a dictionary for information extraction tasks. pages 811–816.

Stephen Soderland. 1999. Learning Information Extraction Rules for Semi-structured and free text. *Machine Learning*, 31(1-3):233–272.

Mark Stevenson and Mark A. Greenwood. 2005. A Semantic Approach to IE Pattern Induction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 379–386, Ann Arbor, MI.

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2001. Automatic Pattern Acquisition for Japanese Information Extraction. In *Proceedings of the Human Language Technology Conference (HLT2001)*.

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2003. An Improved Extraction Pattern Representation Model for Automatic IE Pattern Acquisition. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*, pages 224–231, Sapporo, Japan.

Pasi Tapanainen and Timo Järvinen. 1997. A Non-Projective Dependency Parser. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 64–74, Washington, DC.

Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. 2000. Automatic Acquisition of Domain Knowledge for Information Extraction. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, pages 940–946, Saarbrücken, Germany.

Roman Yangarber. 2003. Counter-training in the Discovery of Semantic Patterns. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*, pages 343–350, Sapporo, Japan.